Estructuras de Datos 1 - ST0245
Examen Parcial 1 - 033 - Lunes

Nombre:.....

Departamento de Informática y Sistemas
Universidad EAFIT

Septiembre 04 de 2017

Criterios de calificación

- Selección múltiple con única respuesta
  - Respuesta correcta: 100 %
  - Respuesta incorrecta: 0 %
- Completar código
  - Respuesta correcta 100 %
  - Respuesta incorrecta o vacía 0 %

NOTAS IMPORTANTES:

- Responda en la hoja de PREGUNTAS
- Marque la hoja de PREGUNTAS

1. Notación O grande 20 %

Para resolver un problema `P(n)`, existen tres algoritmos R, S y T con complejidades $O(n.log(n)+q^2)$, $O(n.log(n)+qn)$ y O $(n.log(n)+q.log(n))$, respectivamente, donde $q = nlog(n)$. De acuerdo a lo anterior, ¿cuál es el orden (**de menor a mayor**) para las complejidades de los algoritmos que resuelven el problema `P(n)`?

A. T, S, R
B. R, S, T
C. S, T, R
D. S, R, T

2. Complejidad 20 %

Considere el siguiente algoritmo

```
1 public int misterio(int n, int i){
2   if (i >= n){
3           return n;
4    }
5     return n * misterio(n, i + 1);
6 }
```

(10 %) ¿Cuántas instrucciones ejecuta el algoritmo en el peor de los casos?

A. $T(n) = T(n-1) + C$
B. $T(n) = T(n-1) + T(n-2) + C$
C. $T(n) = T(n/2) + C$
D. $T(n) = T(n+1) + C$

(10 %) ¿Qué calcula el algoritmo `misterio(n,0)`?

A. El factorial de n
B. n^n
C. n^i
D. La suma de los primeros n números

3. Complejidad 20 %

Sabemos que `P(n)` ejecuta n^3+n pasos y que `D(n)` ejecuta $n+7$ pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función `B(n)`?

```
public int B(int n) {
  int getP = P(n);
```

```
int ni = 0;
for(int i = 0; i < n; ++i){
  if(D(i) > 100){
    ni++;}}
int nj = getP + D(n) * ni;
return nj; }
```

A. $O(n^4)$
B. $O(n^3)$
C. $O(n^2)$
D. $O(2^n)$

4. Recursion 20 %

Alek y Krish están jugando *Número*. Número es un juego en el que un jugador 1, entrega un numero n ($1 <= n <= 10100$) a un jugador 2 y el jugador 2 debe determinar la suma de todos los dígitos de n, exceptuando el caso en el que hay dos dígitos adyacentes (es decir, contiguos, seguidos) que son iguales. Si hay dos dígitos adyacentes, no se suma ninguno de los dos numeros adyacentes. Entre Alek y Krish escribieron un código para hacer esto más rápido, pero se ha borrado una parte. ¿Podrías ayudarles a reconstruir el código a Alek y Krish?

```
1 public int suma(String n) {
2    return sumaAux(n, 0);
3 }
4
5 private int sumaAux(String n, int i){
6    if (i >= n.length()) {
7      return 0;
8    }
9    if(i + 1 < n.length() &&
        n.charAt(i) == n.charAt(i + 1)){
10     return ______;
11   }
12   return (n.charAt(i) - '0') + _____;
13   }
```

La operación `n.charAt(i) - '0'` convierte un caracter en su equivalente en entero, por ejemplo, el caracter `'1'` lo transforma en el número 1.

(10 %) Complete la línea 10.

(10 %) Complete la línea 12.

---

5. Recursión 20 %

Dado un conjunto S de n elementos se quiere determinar si existe un subconjunto R de S tal que la suma de los elementos de R es igual a t, con la condición que si se toma un elemento S_i par, el siguiente elemento S_{i+1} no puede estar en R.

```
1 public boolean comb(int[] S, int i, int t){
2     if(i >= S.length){
3       return t == 0;
4     }
5     //par
6     if(S[i] % 2 == 0){
7       return comb(S, i + 2, t - S[i])
8           || comb(S, i + 1, t); }
9     return comb(___________) ||
10           comb(___________);
11 }
```

Al anterior código le faltan algunas lineas. Completarlas por favor.

(10 %) Linea 9 _____

(10 %) Linea 10 _____